

On the String Kernel Pre-Image Problem with Applications in Drug Discovery

Sébastien Giguère, Amélie Rolland¹, François Laviolette, Mario Marchand

Université Laval

Département d'informatique et de génie logiciel

Abstract

The pre-image problem has to be solved during inference by most structured output predictors. For string kernels, this problem corresponds to finding the string associated to a given input. An algorithm capable of solving or finding good approximations to this problem would have many applications in computational biology and other fields. This work uses a recent result on combinatorial optimization of linear predictors based on string kernels to develop, for the pre-image, a low complexity upper bound valid for many string kernels. This upper bound is used with success in a branch and bound searching algorithm. Applications and results in the discovery of druggable peptides are presented and discussed.

1 Introduction

This work focuses on the challenges of using regression learning algorithms in the design of highly active peptides. Let \mathcal{A} be the set of all amino acids and \mathcal{A}^* be the set of all possible peptides. Throughout this paper we will assume that we have a dataset $\mathcal{S} = \{(\mathbf{x}_1, e_1), \dots, (\mathbf{x}_m, e_m)\} \in \mathcal{A}^* \times \mathbb{R}$ where \mathbf{x}_i is the amino acid sequence of the i -th peptide in \mathcal{S} and e_i is its bioactivity (which could be its binding affinity to some target protein, its antimicrobial activity, or some other desirable activity for peptides). The regression approach consists of learning a predictor h from the training dataset \mathcal{S} . Let $h(\mathbf{x})$ be the estimated bioactivity of \mathbf{x} according to the predictor h . There are many ways to represent such a predictor but this work focuses on string kernel based predictors. Many learning algorithm, such as the Support Vector Regression, Ridge Regression, and Gaussian Processes, produce predictors h whose output $h(\mathbf{x})$, on input \mathbf{x} , is given by

$$h(\mathbf{x}) = \sum_{i=1}^m \alpha_i K(\mathbf{x}_i, \mathbf{x}), \quad (1)$$

where α_i is the weight on the i -th training example, $K(\mathbf{x}_i, \mathbf{x}) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$ is a string kernel, $\langle \cdot, \cdot \rangle$ denotes the inner product between feature vectors, and ϕ is the feature map associated to K . The weight vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_m)$ is obtained by minimizing the learning algorithm objective function.

For the discovery of peptide inhibitors and peptides that could act as drug precursors, we are interested in finding the peptide \mathbf{x}^h maximizing the predicted bioactivity:

$$\mathbf{x}^h = \arg \max_{\mathbf{x} \in \mathcal{A}^*} h(\mathbf{x}). \quad (2)$$

The complexity of this combinatorial problem will obviously depend on the string kernel used to learn h . For few kernels like the Hamming kernel, solving Equation (2) is trivial [1]. However, this kernel is not well suited for peptides [2]. On another hand, the Generic String (GS) kernel [2] is well suited for peptides. It is defined as follows:

$$GS(\mathbf{x}, \mathbf{x}', k, \sigma_p, \sigma_c) \stackrel{\text{def}}{=} \sum_{l=1}^k \sum_{i=0}^{|\mathbf{x}|-l} \sum_{j=0}^{|\mathbf{x}'|-l} \exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right) \exp\left(\frac{-\|\boldsymbol{\psi}^l(x_{i+1}, \dots, x_{i+l}) - \boldsymbol{\psi}^l(x'_{j+1}, \dots, x'_{j+l})\|^2}{2\sigma_c^2}\right), \quad (3)$$

¹Corresponding author: amelie.rolland.1@ulaval.ca

Peer-reviewed and accepted for presentation at Machine Learning in Computational Biology 2014, Montréal, Québec, Canada.

where k controls the length of compared k -mers, $\psi^k : \mathcal{A}^k \rightarrow \mathbb{R}^{dk}$ encodes the physico-chemical properties of k -mers by mapping each of the k amino acids to a real valued vector containing d properties, σ_c controls the penalty incurred when the physico-chemical properties of two k -mers differ, and σ_p controls the penalty incurred when two k -mers are not sharing the same position in their respective peptides. Depending on the chosen hyper-parameters, this kernel can be specialized to eight known kernels [2], namely the Hamming kernel, the Blended Spectrum [3], the Radial Basis Function (RBF), the Oligo [4], and the Weighted degree [5]. Since the proposed approach uses the GS kernel it is also valid for all of these kernels.

It was recently shown [6] that when K is the Generic String (GS) kernel, and when we restrict peptides to be of length l , the peptide $\mathbf{x}^h \in \mathcal{A}^l$ can be found in polynomial time for any predictor h in the form of Equation (1). Their approach maps the combinatorial problem to a directed acyclic graph (DAG) that is basically a *de* Bruijn graph with weights on the arcs. Then, they show that finding the longest (weighted) path in this graph is equivalent to finding $\mathbf{x}^h \in \mathcal{A}^l$. Finally, the longest path can be found in polynomial time by dynamic programming since the graph is acyclic.

However, with the GS kernel, given two peptides \mathbf{x} and \mathbf{x}' of the same length, the Euclidean norms of their feature vectors $\phi(\mathbf{x})$ and $\phi(\mathbf{x}')$ can differ substantially, i.e. we can have $\sqrt{K(\mathbf{x}, \mathbf{x})} \gg \sqrt{K(\mathbf{x}', \mathbf{x}')}$. Let us first show, with a simple example, why this can be problematic. We will then show how to avoid this problem through kernel normalization. Such normalization does not have any computational impact in the learning phase of the regression algorithm. It does however impact substantially the prediction phase, leading to a harder combinatorial problem.

Hence, let us consider the strings “AAAAA” and “ABCDE” and the Spectrum kernel [7] (also known as the n -gram kernel) with k -mers of size two. The string “AAAAA” has a norm of $\sqrt{16} = 4$, while “ABCDE” has a norm of $\sqrt{1+1+1+1} = 2$. Hence the Spectrum kernel is sensitive to k -mers repetitions in the string. Note that this problem is shared by many other string kernels. In the case of the GS kernel, it is possible to avoid this problem by fixing the hyper-parameter σ_p to 0, which forces a constant norm, i.e. $K(\mathbf{x}, \mathbf{x}) = c$ for all $\mathbf{x} \in \mathcal{A}^l$. However, $K(\mathbf{x}, \mathbf{x})$ will vary whenever $\sigma_p > 0$, as it is the case for the Blended Spectrum and the Oligo kernels, for example. A consequence of a non-constant norm is that $h(\mathbf{x})$ will heavily depend on the norm of $\phi(\mathbf{x})$. Hence, \mathbf{x}^h will be more likely a peptide having a feature vector with a large norm, i.e., a peptide having many repetitions. This is generally an undesired bias. It is easy to overcome this problem by normalizing kernel values, in that case the output function becomes

$$h_\star(\mathbf{x}) = \sum_{i=1}^m \alpha_i \frac{K(\mathbf{x}_i, \mathbf{x})}{\sqrt{K(\mathbf{x}_i, \mathbf{x}_i)K(\mathbf{x}, \mathbf{x})}} = \frac{1}{\sqrt{K(\mathbf{x}, \mathbf{x})}} \sum_{i=1}^m \beta_i K(\mathbf{x}_i, \mathbf{x}) \quad (4)$$

where $\beta_i = \frac{\alpha_i}{\sqrt{K(\mathbf{x}_i, \mathbf{x}_i)}}$. In that case we are now interested in finding

$$\mathbf{x}^{h_\star} = \arg \max_{\mathbf{x} \in \mathcal{A}^*} \frac{1}{\sqrt{K(\mathbf{x}, \mathbf{x})}} \sum_{i=1}^m \beta_i K(\mathbf{x}_i, \mathbf{x}). \quad (5)$$

This optimization problem is also a pre-image problem, but written in a slightly different form. We conjecture that solving Equation (5) is \mathcal{NP} -Hard when K is the GS kernel. Given the similarity between the problems of Equation (2) and Equation (5), the difference in their computational complexity is unexpected.

In the next section, we will present a low complexity upper bound on Equation (4) that makes it a good candidate for a branch and bound search to solve Equation (5).

2 Method

Since the number of peptides grows exponentially with its length, it becomes impossible to evaluate all solutions for large peptides, we propose a branch and bound scheme to guide this search. A branch and bound algorithm starts by dividing the search space into disjoint subspaces. For example, one subspace could be all peptides ending with the string “DE”. For a maximization problem, an upper bound on the best achievable solution is computed for each of these subspaces. Then, the subspace with the highest upper bound is further divided into smaller subspaces. Finally, the search stops when a subspace can no longer be divided (a leaf is reached in the search tree), or when the upper bound value is lower than the value of an already achieved solution (*i.e.*, an already reached leaf in the search tree). A branch and bound approach can thus avoid exploring a large part of the search space.

Algorithm 1 gives the specifics of the branch and bound algorithm applied to our case. The search algorithm alternates between a greedy phase and a branch and bound phase. The greedy phase is important to ensure that leaves of the search tree are quickly visited. This allows good but sub-optimal solutions to be returned by the algorithm if the allowed computational time expires. Whenever a node is visited, the bound is computed for all its children and they

are added to the priority queue accordingly. This greedy process is repeated until a leaf is reached. Then, the node with the largest bound is visited and the greedy process starts again. At all time, the best solution found so far is kept and the search stops when the bound of the node on top of the priority queue is smaller than the value of the best solution.

Algorithm 1 Branch and bound search for maximal string of length l

\mathcal{Q} : empty priority queue ordering bounds in descending order

```

best_node ← Node(empty_string, 0)
for all  $s \in \mathcal{A}^k$  do                                ▷ Add all  $k$ -mer in  $\mathcal{Q}$ 
     $\mathcal{Q}.push(Node(s, F(s, l)))$ 
end for
while  $current\_node \leftarrow \mathcal{Q}.pop()$  &  $current\_node.bound > best\_node.bound$  do    ▷ Get maximal node in  $\mathcal{Q}$ 
    while  $|current\_node.string| < l$  &  $current\_node.bound > best\_node.bound$  do
        best_child ← Node(empty_string, 0)
        for all  $a \in \mathcal{A}$  do                                ▷ Evaluate all children of node
             $s' \leftarrow Concatenate(a, current\_node.string)$ 
            if  $F(s', l) > best\_node.bound$  then
                if  $F(s', l) > best\_child.bound$  then
                    best_child ← Node( $s', F(s', l)$ )        ▷ Update best_child
                end if
                 $\mathcal{Q}.push(Node(s', F(s', l)))$                 ▷ Add child to  $\mathcal{Q}$ 
            end if
        end for
        current_node ← best_child
         $\mathcal{Q}.remove(best\_child)$                             ▷ Remove best_child from  $\mathcal{Q}$  if it was added
    end while
    if  $|current\_node.string| = l$  &  $current\_node.bound > best\_node.bound$  then    ▷ Update best_node
        best_node ← current_node
    end if
end while

return best_node.string, best_node.bound                ▷ Return string and maximal bioactivity

```

Let $\mathcal{A}^{l-p} \times \{x'_1, \dots, x'_p\}$ be the set of all possible strings of length l that end with x'_1, \dots, x'_p , in other words, $\mathcal{A}^{l-p} \times \{x'_1, \dots, x'_p\}$ is a set of strings having their last p characters fixed. Our goal is to have a function F that upper bounds h_* for every $\mathcal{A}^{l-p} \times \{x'_1, \dots, x'_p\}$. In other words:

$$F(\mathbf{x}', l) \geq \max_{\mathbf{x} \in \mathcal{A}^{l-p} \times \{x'_1, \dots, x'_p\}} \frac{1}{\sqrt{K(\mathbf{x}, \mathbf{x})}} \sum_{i=1}^m \beta_i K(\mathbf{x}_i, \mathbf{x}). \quad (6)$$

To do so, let $F(\mathbf{x}', l) \stackrel{\text{def}}{=} \frac{1}{\sqrt{f(\mathbf{x}', l)}} g(\mathbf{x}', l)$, where

$$f(\mathbf{x}', l) \leq \min_{\mathbf{x} \in \mathcal{A}^{l-p} \times \{x'_1, \dots, x'_p\}} K(\mathbf{x}, \mathbf{x}) \quad \text{and} \quad g(\mathbf{x}', l) \geq \max_{\mathbf{x} \in \mathcal{A}^{l-p} \times \{x'_1, \dots, x'_p\}} \sum_{i=1}^m \beta_i K(\mathbf{x}_i, \mathbf{x}) \quad (7)$$

are, respectively, a lower and an upper bound.

When K is the GS kernel with hyper-parameters k , σ_p and σ_c , the lower bound f can be obtained as follows:

$$f_{k, \sigma_p, \sigma_c}(\mathbf{x}', l) \stackrel{\text{def}}{=} GS(\mathbf{x}', \mathbf{x}', k, \sigma_p, \sigma_c) + 2XX'(\mathbf{x}', l, k, \sigma_p, \sigma_c) + XX(\mathbf{x}', l, k, \sigma_p, \sigma_c), \quad (8)$$

where

$$XX'(\mathbf{x}', l, k, \sigma_p, \sigma_c) \stackrel{\text{def}}{=} \sum_{p=1}^k \sum_{i=1}^{l-|\mathbf{x}'|} \min_{\mathbf{x} \in \mathcal{A}^p} \sum_{j=1}^{|\mathbf{x}'|} \exp\left(\frac{-(i-j)^2}{2\sigma_p^2}\right) \exp\left(\frac{-\|\boldsymbol{\psi}^l(\mathbf{x}_1, \dots, \mathbf{x}_p) - \boldsymbol{\psi}^l(x'_j, \dots, x'_{j+p})\|^2}{2\sigma_c^2}\right), \quad (9)$$

$$XX(\mathbf{x}', l, k, \sigma_p, \sigma_c) \stackrel{\text{def}}{=} \sum_{p=1}^k \sum_{i=1}^{l-|\mathbf{x}'|} \sum_{j=1}^{l-|\mathbf{x}'|} \exp\left(\frac{-(i-(l-|\mathbf{x}'|+j))^2}{2\sigma_p^2}\right) \exp\left(\frac{-(D(i, j)^2 + \dots + D(i+p, j+p)^2)}{2\sigma_c^2}\right) \quad (10)$$

and

$$D(i, j) = \begin{cases} 0 & \text{if } i = j, \\ \max_{a, a' \in \mathcal{A}} \psi(a, a') & \text{otherwise.} \end{cases}$$

Note that the lower bound f is not attained since it under-estimates the value of the string $\mathbf{x} \in \mathcal{A}^{l-p} \times \{x'_1, \dots, x'_p\}$ minimizing $K(\mathbf{x}, \mathbf{x})$.

For $g(\mathbf{x}', l)$, note that $\sum_{i=1}^m \beta_i K(\mathbf{x}_i, \mathbf{x})$ is what [6] proposed to maximize. Their approach uses a dynamic programming table to compute the longest path in a graph. It is relatively easy to modify their algorithm to return the table instead of the longest path. In that way, given a string with suffix \mathbf{x}' , it is possible to determine in constant time, by accessing the dynamic programming table, the prefix from \mathcal{A}^{l-p} maximizing $\sum_{i=1}^m \beta_i K(\mathbf{x}_i, \mathbf{x})$. The computation of g is thus very efficient, the algorithm of [6] only needs to be done once before the branch and bound search, then $g(\mathbf{x}', l)$ can be computed in constant time for any \mathbf{x}' . Finally, g is the least upper bound (or suprema) since there is always a string $\mathbf{x} \in \mathcal{A}^{l-p} \times \{x'_1, \dots, x'_p\}$ with $g(\mathbf{x}, l) = \sum_{i=1}^m \beta_i K(\mathbf{x}_i, \mathbf{x})$. In other words, there are no tighter bound.

3 Results and Discussion

We followed the protocol suggested by [6] and, as a proof of concept, we used the same dataset they used: 101 cationic antimicrobial pentadecapeptides (CAMPs) from the synthetic antibiotic peptides database [8]. Peptide antibacterial activities are expressed as the logarithm of bactericidal potency. As in [6], we used kernel ridge regression as the learning algorithm. Except when stated otherwise, all hyper-parameters for the GS kernel (k, σ_c, σ_p) and the kernel ridge regression (λ) were chosen by standard cross-validation. We learned two predictors of antimicrobial potency, one uses unnormalized kernel values (thus, the same predictor used in [6]), the other was trained using normalized kernel values. We refer to these predictors respectively as h and h_* .

The method presented in [6] was used to identify the peptide $\mathbf{x}^h \in \mathcal{A}^{15}$ of maximal predicted bioactivity according to h and the branch and bound was used to identify $\mathbf{x}^{h_*} \in \mathcal{A}^{15}$, the peptide maximizing h_* . Both approaches found the same peptide, which is “WWKWWKRLRLFLLV”.

Note that both methods are able to output more than one sequence. The method of [6] uses a k -longest path algorithm to obtain the k peptides of maximal bioactivity. It is also possible for the branch and bound by stopping the search only when the bound on top of the priority queue is lower than the k -th peptide found. These peptides can be used, for example, for motif generation, for multiple peptide synthesis or in combinatorial chemistry. To highlight the differences between the methods, they were used to list the top 1000 peptides and the lists were compared: 70.5% of the 1000 peptides were found by both methods. Then, the Pearson correlation coefficient (PCC) was computed between the rank of the 679 peptides present in both lists: a PCC of 0.56 was obtained. Hence, the ranking of peptides found by h and by h_* differ significantly.

The overlap in the lists is attributed to the value of σ_p that was chosen during cross-validation for h . As explained earlier, when σ_p is 0, the unnormalized predictor h will not suffer from differences in the norm of $\phi(\mathbf{x})$. In the case of antimicrobial peptides, $\sigma_p = 0.8$ was found to be the optimal value for h . This suggests that the method of [6] offers some resistance to variation in norm when σ_p is small. Indeed, for small σ_p , all examples have about the same norm.

To further highlight the difference between the methods, we intentionally fixed σ_p to infinity and cross-validated all other parameters for h and h_* and compared the best peptides found. Situations where σ_p would have to be set to infinity are not at all unlikely. For example, cyclic peptides have no N-terminus and no C-terminus. For that reason, there is no origin from which we can express the positions of k -mers in these peptides. The method of [6] found the peptide “FKKIFKKIFKKIFKF” using the predictor h and the branch and bound approach found the peptide “WKKIFKKIWKFVRVK” using the predictor h_* . The peptide identified by h shares almost no similarity with peptides of the training set and is basically composed of repetition of the k -mer “FKK”. In contrast, the peptide identified by h_* shares many substructures with the most bioactive peptides of the training set. This tends to point out that in situation where example norms vary a lot, h clearly favors examples having a large norm. However, this bias is unjustified and is not related to a biological reality.

4 Conclusion

We proposed a bound for maximizing the inference function of kernel methods that uses normalized string kernels. Moreover, the bound is also valid for solving the pre-image of a variety of string kernels. Empirical results show that the method proposed by [6] can suffer from a dominance of the norm for certain strings, a problem which is present

with many string kernels. In these situations, the proposed method was shown to overcome this problem. Tighter bounds should take advantage of the proposed framework and allow the discovery of novel peptide inhibitors. Finally, applications for drugs based on cyclic peptides and other structured output applications are expected.

Acknowledgments

The authors would like to thank Prudencio Tossu for his help in the experimentations. Computations were performed on the Colosse supercomputer at Université Laval (resource allocation project: avt-710-aa), under the auspices of Calcul Québec and Compute Canada. AR is recipient of a Master's Scholarship from the Fonds de recherche du Québec - Nature et technologies (FRQNT). This work was supported by the FRQNT (FL & MM; 2013-PR-166708).

References

- [1] Sébastien Giguère, François Laviolette, Mario Marchand, and Khadidja Sylla. Risk bounds and learning algorithms for the regression approach to structured output prediction. In *International Conference on Machine Learning (ICML)*, 2013.
- [2] Sébastien Giguère, Mario Marchand, François Laviolette, Alexandre Drouin, and Jacques Corbeil. Learning a peptide-protein binding affinity predictor with kernel ridge regression. *BMC Bioinformatics*, 14, 2013.
- [3] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.
- [4] P. Meinicke, M. Tech, B. Morgenstern, and R. Merkl. Oligo kernels for datamining on biological sequences: A case study on prokaryotic translation initiation sites. *BMC Bioinformatics*, 5, 2004.
- [5] Gunnar Rätsch and Sören Sonnenburg. Accurate Splice Site Detection for *Caenorhabditis elegans*. In B and J. P. Vert, editors, *Kernel Methods in Computational Biology*, pages 277–298. MIT Press, 2004.
- [6] Sébastien Giguère, François Laviolette, Mario Marchand, Denise Tremblay, Sylvain Moineau, Éric Biron, and Jacques Corbeil. Machine learning assisted design of highly active peptides for drug discovery. *Under review, Submitted to MLCB*, 2014.
- [7] Christina S Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific symposium on biocomputing*, volume 7, pages 566–575. World Scientific, 2002.
- [8] David Wade and Jukka Englund. Synthetic antibiotic peptides database. *Protein and peptide letters*, 9(1):53–57, 2002.